Optimizing I/O Costs of Multi-dimensional Queries using Bitmap Indices

Doron Rotem, Kurt Stockinger, Kesheng Wu

Lawrence Berkeley National Laboratory
University of California
1 Cyclotron Road, Berkeley, California 94720, USA
{D_Rotem, KStockinger, KWu}@lbl.gov

Abstract. Bitmap indices are efficient data structures for processing complex, multi-dimensional queries in data warehouse applications and scientific data analysis. For high-cardinality attributes, a common approach is to build bitmap indices with binning. This technique partitions the attribute values into a number of ranges, called bins, and uses bitmap vectors to represent bins (attribute ranges) rather than distinct values. In order to yield exact query answers, parts of the original data values have to be read from disk for checking against the query constraint. This process is referred to as *candidate check* and usually dominates the total query processing time.
In this paper we study several strategies for optimizing the *candidate check* cost for multi-dimensional queries. We present an efficient *candidate check* algorithm based on attribute value distribution, query distribution as well as query selectivity with respect to each dimension. We also show that re-ordering the dimensions during query evaluation can be used to reduce I/O costs. We tested our algorithm on data with various attribute value distributions and query distributions. Our approach shows a significant improvement over traditional binning strategies for bitmap indices.

1 Introduction

Large-scale data analysis of data warehouses and scientific applications requires efficient index data structures to cope with the increasing size and complexity of data. Bitmap indices are often used for querying large, multi-dimensional, read-only data stores. Due to its efficiency, this technique was also implemented by the major commercial database vendors.

The simplest form of bitmap indices works well for low-cardinality attributes, such as "gender", "types of cars sold per month", or "airplane models produced by Airbus and Boeing". However, for high-cardinality attributes such as "distinct temperature values in a supernova explosion", simple bitmap indices are impractical due to large storage and computational complexities. In this case, bitmap indices are built on attribute ranges (bins) rather than on distinct attribute values. The advantage of this approach is that a lower number of bitmap

vectors is required. On the other hand, parts of the original data (candidates) have to be read from disk in order to get exact query answers. This process is called *candidate check*.

An example of a bitmap index with bins is given in Figure 1. Assume that we want to evaluate the query $37 \leq x < 63$. Bins 1, 2 and 3 contain the relevant data values. However, Bins 1 and 3 are edge bins since they contain also irrelevant values. Answering this query involves checking the values on disk corresponding to the four "1-bits" in these two columns. In this example only one of the four values qualifies, namely, 61.7. We call this additional step the *candidate check*. As we can see from this example, the cost of performing a *candidate check* on an edge bin is related to the number of "1-bits" in that bin.

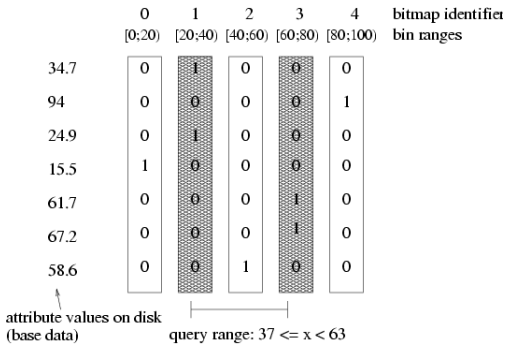

Fig. 1. Range query $37 \leq x < 63$ on a bitmap index with binning.

In previous work we have shown that the *candidate check* is the major bottleneck of bitmap indices with binning. In this paper we use a dynamic programming algorithm, called *Dynamic-Bin*, for optimizing one-dimensional queries presented in [10] to achieve an efficient multi-attribute binning strategy. The key idea in *Dynamic-Bin* is to use query workload and data value distribution statistics in order to calculate the optimal location of bin boundaries. This is done by placing relatively more bins in regions of the data "heavily hit" by queries or containing a large fraction of the data.

The main contributions of this paper are:

- We study optimization issues related to multi-dimensional queries with bitmap indices. We show that the *candidate check* is the dominant part in query evaluation and introduce an optimization strategy based on attribute value distribution, query distribution as well as query selectivity with respect to each dimension.
- We show that re-ordering the dimensions during query evaluation can be used to further reduce I/O costs.

– We provide detailed experimental results on data with various attribute value distributions and query distributions. The results demonstrate a significant improvement over traditional binning strategies for bitmap indices.

2 Related Work

Bitmap indices are used for accelerating complex, multi-dimensional queries for On-Line Analytical Processing and data warehouses [2] as well as for scientific applications [11]. They were first implemented in a commercial DBMS called Model 204 [8]. Improvements on this approach were discussed in [9].

Various bitmap encoding strategies for low-cardinality attributes are presented in [1, 14]. In order to overcome the storage complexity of bitmap indices, bitmap compression algorithms were evaluated in [4]. More recently a new compression scheme called Word-Aligned Hybrid (WAH) [12] was introduced. This compression algorithm significantly reduces the overall query processing time compared to existing algorithms.

A binning scheme for bitmap indices on high-cardinality attributes was discussed in [13]. This idea was extended and successfully applied for large-scale scientific data [11]. The authors demonstrated that bitmap indices with binning can significantly speed up multi-dimensional queries on high-cardinality attributes.

In [5] a methodology for building space efficient bitmap indices is introduced for high-cardinality attributes based on binning. The work in [5] focuses on point (equality) queries rather than range queries discussed in this paper. Similar to our approach, an optimal dynamic programming algorithm is used for efficiently choosing bin ranges. Our approach greatly reduces the complexity of the algorithm by proving that only query endpoints need to be considered as potential locations for bin boundaries rather than all possible values of the attribute as in [5].

The literature on histograms is partially related to bitmap indices. The optimal construction of range histograms is discussed in [6, 3]. The main difference is that for bitmap indices precise answers are required and therefore the objective is to minimize disk access costs to edge bins. However, in the histogram case, some statistical techniques can be used to estimate errors without actual access to original data on disk.

3 Preliminaries

In order to make the paper self-contained, we summarize here our results for optimizing the candidate check for a single attribute. Detailed proofs can be found in [10]. Assume a dataset D has N records with a single attribute A. For simplicity we will assume that each value of the attribute is an integer in the range $[1, n]$. We are also given a collection of range queries Q such that each $q \in Q$ defines a range $q = [l_q, u_q)$ open on the right (i.e., it includes the points $l_q, l_q + 1, ..., u_q - 1$) and is associated with a probability p_q reflecting its relative

popularity. The points $l_q \in [1, n]$ and $u_q \in [2, n+1]$ are called endpoints of query q**.** A bitmap index on A is built by partitioning the range $[1, n]$ into bins with one bitmap (consisting of N bits) associated with each bin as previously described. An integer constraint k, specifies the maximum number of bins allowed, i.e., it is required to partition the range $[1, n]$ into k successive sub-ranges (bins) $B =< b_1, b_2, ..., b_k >$. This is done by choosing $k-1$ integer bin boundary points x_i where $1 < x_1 < x_2 < ... < x_{k-1} < n + 1$. The sub-ranges associated with bins b_i are all open on the right and defined as follows:

$$b_1 = [1, x_1)$$
$$b_i = [x_{i-1}, x_i) \text{ for } 2 \leq i \leq k$$
$$b_k = [x_{k-1}, n + 1)$$

A bin $b \in B$ is defined as an edge bin for query q if the range defined by the query q overlaps some part of the range defined by bin b but not its whole range i.e., $q \cap b \neq \emptyset$ and $q \cap b \neq b$. In general, a query may have 0, 1, or 2 edge bins.

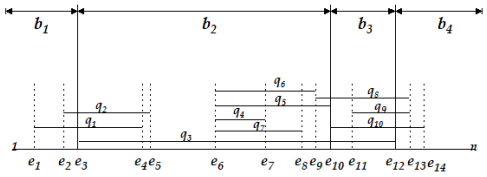

Fig. 2. Query endpoints and bin boundaries. Horizontal lines represent query ranges. Dotted vertical lines mark query endpoints.

In Figure 2 a set of 10 range queries and a binning into 4 bins is shown. In this example query q_3 has no edge bins since both its endpoints fall on bin boundaries. Each of the queries q_4, q_5, q_6, q_7, q_{10} has 1 edge bin and each of the queries q_1, q_2, q_8, q_9 have 2 edge bins. As explained earlier, when query q is specified, a significant fraction of the I/O costs it incurs is related to the number of data pages we need to read in order to perform *candidate check* on each of its edge bins. For a given bin b, let $E(b)$ denote the set of queries that have bin b as an edge bin. For example, in Figure 2 $E(b_1) = \{q_1, q_2\}$; $E(b_2) = \{q_1, q_2, q_4, q_5, q_6, q_7, q_8\}$; $E(b_3) = \{q_9\}$; $E(b_4) = \{q_8, q_9, q_{10}\}$.

Let n_b denote the number of data values that fall into the range defined by b, this is also the number of "1-bits" in the bitmap corresponding to b. Based on the usual assumption that records are distributed uniformly across pages and assuming that the total number of pages occupied by attribute A is P, the expected number of disk pages that contain data values that fall in the range defined by bin b denoted by P_b, satisfies [9].

$$P_b = P(1 - (1 - \frac{1}{P})^{n_b}) \approx P(1 - e^{-\frac{n_b}{P}}) \quad (1)$$

The expected I/O cost of answering the queries in Q when an attribute range is partitioned by the set of bins B is defined as

$$Cost(Q, B) = \sum_{b \in B} P_b \sum_{q \in E(b)} p_q \tag{2}$$

The inner sum computes the total probability of all the queries that use a given bin b as an edge bin. This is then multiplied by the I/O cost of the bin (expected number of pages) and summed over all bins.

The problem we wish to solve, *OptBin* is defined as follows:

Given a dataset D with one attribute, a set of range queries Q and a constraint k on the number of bins, find a binning B^k_{opt} of the attribute range [1,n] into k bins that minimizes the total I/O cost of candidate check.

4 The Multi-Attribute Candidate Check Problem

In this section we present results for the multi-attribute *candidate check* problem and its relationship to the single attribute case. We will start with some definitions. Let D be a dataset with N records defined over t attributes $A_1, A_2, ..., A_t$. Each record $R \in D$ has the form $R = v_1, v_2, ..., v_t$ where v_i represents its value with respect to attribute A_i. Let us assume that the range of possible values for each attribute A_i is $[1, n]$. A set Q of multi-attribute range queries is given where a query $q \in Q$ defines an intersection of t ranges and has the form $q = \bigcap_{i=1}^{t} r_i$ where $r_i = [l^i_q, u^i_q)$ defines the range of permissible values for attributeA_i (a range r_i is commonly omitted from q in the trivial case that it includes all permissible values of the attribute, i.e., $l^i_q = 1$ and $u^i_q = n$). A record $R = v_1, v_2, ..., v_t$ satisfies the range r_i if the value v_i falls in this range, i.e., $v_i \in [l^i_q, u^i_q)$. It satisfies the query q if it satisfies all its ranges, i.e., $v_i \in [l^i_q, u^i_q)$ for$1 \leq i \leq t$.

In order to get a handle on the issues involved with multi-attribute candidate check problem, let us first assume a bitmap index for each attribute was constructed according to some binning strategy. Given a query q and such a collection of bitmap indices, each range r_i defines 0, 1 or 2 edge bins in its respective bitmap index. A simple algorithm for answering a query q, which we call *Simple-CC*, is to perform an independent *candidate check* algorithm for each range $r_i = [l^i_q, u^i_q)$ using the bitmap index built for attribute A_i and present the result in a bitmap $b(r_i)$ with N entries. A "1-bit" in position j of $b(r_i)$ represents the fact that the j^{th} record in D satisfies $[l^i_q, u^i_q)$. This is then followed by performing a Boolean AND operation on all the $b(r_i)$'s to obtain the final result. The algorithm *Simple-CC* will require accessing each value in the edge bins of all attributes. Note that the cost for *Simple-CC* is independent of the order in which the candidate checks are performed on the various attributes.

The *Simple-MultiOptBin* (SMOB) problem which generalizes *OptBin* is defined as follows:

Given a multi-dimensional dataset D, a set of range queries Q and a constraint k on the total number of bins, find t integers $k_1, k_2, ..., k_t$ where $k =$

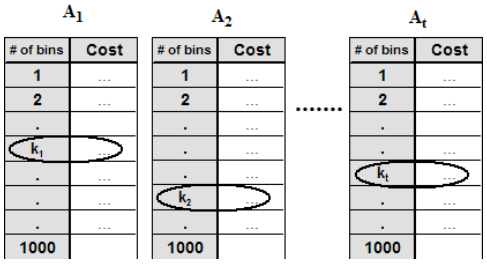

Fig. 3. Multi-attribute bin selection

$\sum_{i=1}^{t} k_i$ *and locations for bin boundaries such that* k_i *bins are allocated for the bitmap index for attribute* A_i *and the total expected I/O cost of candidate check using Simple-CC is minimized.*

We can show that for each fixed selection of t integers $k_1, k_2, ..., k_t$ where $k = \sum_{i=1}^{t} k_i$ and k_i bins are allocated to attribute A_i, a solution in polynomial time $O(tkr^2)$ can be constructed by applying *Dynamic-Bin* algorithm separately for each attribute. This is done as follows:

Consider the set of queries q_i obtained from Q by taking from each query $q \in Q$ only its range relating to attribute A_i (i.e, $q_i = \bigcup_{q \in Q}[l_q^i, u_q^i)$). As the total amount of I/O cost in answering q is the sum of I/O costs incurred by answering each q_i, *Dynamic-Bin* algorithm is executed with constraint k_i on the number of bins and considering only queries in q_i.

The SMOB problem for $k = 1000$ is illustrated in Figure 3 where each table represents the output from applying the *Dynamic-Bin* algorithm on a single attribute and selecting some k_i for each attribute. The main problem is determining the values of the k_is. Unfortunately this turns out to be an NP-hard problem as shown in the next theorem.

Theorem 1. *The SMOB is NP-Hard even if all queries in* Q *have equal probability and each query includes a range for only one attribute.*

Proof. (Outline): The reduction is from a known NP-hard problem called "the multiple-choice knapsack problem" (MCKP) [7]. In the MCKP we are given t groups, each consisting of multiple items where item j in group i has value $v_{i,j}$ and cost $c_{i,j}$. It is required to select exactly one item from each group such that the total cost of selected items does not exceed a budget B and their total value is maximized. Given an instance of MCKP we can transform it to an instance of SMOB where each attribute represents a group with k members, each member represents a choice for the number of bins for that attribute. The values and costs in MCKP can be transformed to candidate check costs and the number of bins used respectively. Solving SMOB with total budget k represents a solution to the MCKP instance. □

Several effective heuristic strategies are known for obtaining sub-optimal solutions for the MCKP problem that are also applicable to the SMOB problem.

In this paper we will not study this problem any further but rather focus on efficient strategies of evaluating queries for a given binning selection.

5 Query Evaluation with Attribute Reordering

An efficient query evaluation strategy is discussed in [11]. It attempts to reduce the amount of I/O costs by performing a *candidate check* algorithm in t phases. The idea is to reduce the I/O costs of accessing edge bins by only retrieving records that survived previous phases. In phase 1 we perform a *candidate check* for range r_1 and produce the bitmap $b(r_1)$. In phase 2 we first perform a Boolean AND between $b(r_1)$ and all potential bitmaps corresponding to the range r_2. We therefore reduce the number of "1-bits" in the edge bins corresponding to r_2. In general, in phase $i+1$ we perform edge bin access only on values corresponding to records that survived the *candidate check* in phase i. The next theorem shows that the I/O cost of this strategy depends on the order of performing the candidate check on the attributes.

Theorem 2. *Given a query $q = \bigcap_{i=1}^{t} r_i$ assume the I/O cost involved in candidate checking for range r_i is W_i and the fraction of records satisfying this range is s_i (selectivity).*

We assume that for all i , $0 < s_i < 1$ thus omitting the trivial cases where for some range r_i either $s_i = 0$ (the query has empty results) or $s_i = 1$ (no candidate check needed for r_i as all values qualify). Let $g_i = \frac{W_i}{1-s_i}$, then the optimal order of candidate check evaluation is in sorted non-decreasing order of g_is.

Proof. (Outline): We will use the notation $S_j = \prod_{i=1}^{j} s_i$. We show that any evaluation order that violates the above order cannot be optimal. Assume some optimal evaluation order has cost C_{opt} and renumber attributes according to that order. The cost of this evaluation is

$$C_{opt} = W_1 + W_2 S_1 + W_3 S_2 + ...W_j S_{j-1} + ... + W_t S_{t-1}$$

This cost expression assumes that in each phase the number of values in the edge bins that need to be checked are reduced by the product of the selectivities from previous phases and the number of disk accesses is approximately linear with the number of records in a bin. Assume that for two consecutive candidate checks in phases j and $j+1$ the sorting order is not obeyed, i.e., $g_j > g_{j+1}$. We will switch the evaluation order between these ranges to obtain another evaluation order with cost C^*, the difference in costs is

$$\begin{aligned} & C^* - C_{opt} \\ & = (W_{j+1} S_{j-1} + W_j s_{j+1} S_{j-1}) - (W_j S_{j-1} + W_{j+1} S_j) \\ & = W_{j+1} S_{j-1}(1 - s_j) - W_j S_{j-1}(1 - s_{j+1}) \\ & = S_{j-1}(1 - s_j)(1 - s_{j+1})(g_{j+1} - g_j) < 0 \end{aligned}$$

The inequality on the last line follows from the fact that each of the first three terms in the product is positive and the last term is negative due to the assumption that $g_j > g_{j+1}$. But this contradicts the optimality of C_{opt} as we found an order with a smaller cost. □

In Section 6 we compare the optimal order of evaluation (based on non-decreasing order of g_is) to three other orders: *alphabetic* which does not take into account any query or data characteristics (non-decreasing alphabetic order by name of attribute), *selectivity* based only on selectivity of each range (non-decreasing order of the s_is) and *candidates* based only on the I/O cost for candidate check (non-decreasing order of the W_is).

6 Experimental Results

In this section we present a representative subset of our experiments to evaluate the efficiency of our new binning and query evaluation strategies. We generated 100 million data points that follow a Zipf distribution with the parameters z=0, 0.5, 1 and 2. For all our experiments we used equality encoded bitmap indices and WAH compression [12]. We also generated 5,000 random range queries. The goal is to compare the following three different binning strategies: a) *Equi-width* binning: Each bin has the same width. b) *Equi-depth* binning: The bin boundaries are chosen in such a way that all bins have roughly the same number of entries. c) *Opt-binning*: The bin boundaries are chosen based on *Dynamic-Bin* introduced in Section 3.

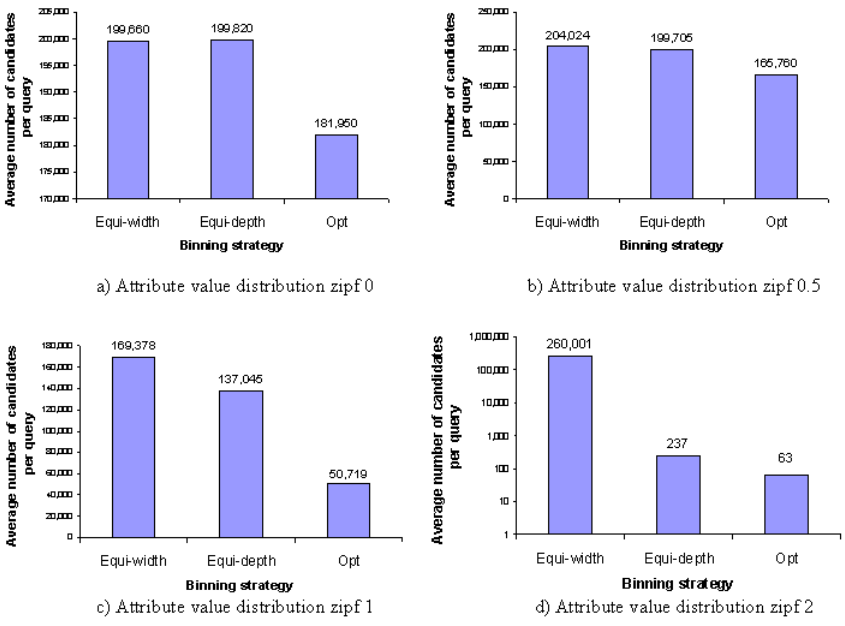

Fig. 4. I/O costs of candidate check per attribute.

Figure 4 shows the average number of candidates per attribute for the 5000 4-dimensional queries. Note that for each attribute the shape of the Zipf distribution is different. As we can see, in all cases, *Opt-binning* outperforms the other two binning strategies. In addition, the relative efficiency of *Opt-binning* increases with more skewness in the data.

The average number of candidates for all four attributes combined is given in Figure 5. We also show the impact of query reordering as discussed in Section 5. We can observe that ordering according to the number of candidates in the edge bins is quite competetive with the optimal reordering strategy for this data. Again we can observe that *Opt-binning* outperforms the other two binning strategies by a factor of 3 for uniform queries. For left-skewed queries the I/O costs are improved by nearly a factor of 4.

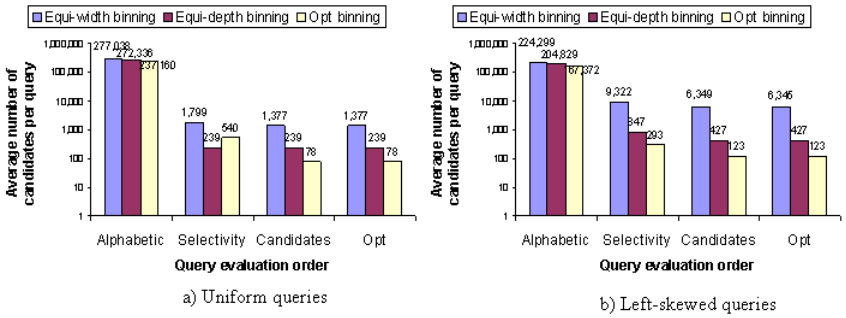

Fig. 5. Combined I/O costs of 4-dimensional queries with different distributions..

7 Conclusions and Future Work

For high-cardinality attributes bitmap indices with binning have a lower storage and computational complexity than simple bitmap indices. However, this advantage comes with an additional cost, the so-called *candidate check* costs for verifying parts of the data against the query constraints. In this paper we studied issues related to optimizing multi-dimensional queries on bitmap indices with bins. We introduced an optimization strategy based on attribute value distribution, query distribution as well as query selectivity with respect to each dimension. Our experimental results on data with various attribute value distributions and query distributions demonstrated that our new algorithm significantly improves the candidate check costs when compared to traditional strategies by at least a factor of 3. We also showed that the efficiency of our algorithm is more significant for highly-skewed data and queries.

In the future we plan to analyze several heuristics that efficiently determine the optimal number of bins for multiple attribute datasets. This problem is very

important for multi-attribute queries where each attribute has different characteristics in terms of data distribution, selectivity and probability of occurring in a query expression.

References

1. C. Y. Chan and Y. E. Ioannidis. An Efficient Bitmap Encoding Scheme for Selection Queries. In *SIGMOD*, Philadelphia, Pennsylvania, USA, June 1999. ACM Press.
2. S. Chaudhuri and U. Dayal. An Overview of Data wharehousing and OLAP Technology. *ACM SIGMOD Record*, 26(1):65–74, March 1997.
3. S. Guha, N. Koudas, and D. Srivastava. Fast Algorithms For Hierarchical Range Histogram Construction. In *PODS 2002*, Madison, Wisconsin, USA, June 2002. ACM Press.
4. T. Johnson. Performance Measurements of Compressed Bitmap Indices. In *International Conference on Very Large Data Bases*, Edinburgh, Scotland, September 1999. Morgan Kaufmann.
5. N. Koudas. Space Efficient Bitmap Indexing. In *International Conference on Information and Knowledge Management*, McLean, Virginia, USA, November 2000. ACM Press.
6. N. Koudas, S. Muthukrishnan, and D. Srivastava. Optimal Histograms for Hierarchical Range Queries. In *PODS*, Dallas, Texas, USA, 2000. ACM Press.
7. A. E. Mohr. Bit Allocation in Sub-linear Time and the Multiple-Choice Knapsack Problem. In *Data Compression Conference*, Snao Bird, Utah, USA, March 2002. IEEE Computer Society Press.
8. P. O'Neil. Model 204 Architecture and Performance. In *2nd International Workshop in High Performance Transaction Systems*, Asilomar, California, USA, 1987. Springer-Verlag.
9. P. O'Neil and D. Quass. Improved Query Performance with Variant Indexes. In *Proceedings ACM SIGMOD International Conference on Management of Data*, Tucson, Arizona, USA, May 1997. ACM Press.
10. D. Rotem, K. Stockinger, and K. Wu. Efficient Binning for Bitmap Indices on High-Cardinality Attributes. Technical report, Berkeley Lab, November 2004.
11. K. Stockinger, K. Wu, and A. Shoshani. Evaluation Strategies for Bitmap Indices with Binning. In *International Conference on Database and Expert Systems Applications (DEXA)*, Zaragoza, Spain, September 2004. Springer-Verlag.
12. K. Wu, E. J. Otoo, and A. Shoshani. On the Performance of Bitmap Indices for High Cardinality Attributes. In *International Conference on Very Large Data Bases*, Toronto, Canada, September 2004. Morgan Kaufmann.
13. K.-L. Wu and P.S. Yu. Range-Based Bitmap Indexing for High-Cardinality Attributes with Skew. Technical report, IBM Watson Research Center, May 1996.
14. M.-C. Wu and A. P. Buchmann. Encoded Bitmap Indexing for Data Warehouses. In *International Conference on Data Engineering*, Orlando, Florida, USA, February 1998. IEEE Computer Society Press.